CLAIMS

What is claimed is:

1. A process for requesting authentication comprising:

transmitting data from a hash digest formed using client-specific data together with second client specific data; and

receiving, in response to transmitting, an indication of acceptance when the data from the hash digest corresponds to a valid client authentication request.

2. The process of claim 1, further comprising, prior to transmitting, computing the hash digest using first client specific data comprising a valid client name together with a time-varying function.

3. The process of claim 1, further comprising, prior to transmitting, computing a hash digest using an HMAC algorithm and wherein the data from the hash digest include a truncated hash digest.

4. The process of claim 1, further comprising, prior to transmitting, computing a hash digest using the client name, client key and a function of time, and wherein transmitting includes transmitting a current time.

5. The process of claim 1, further comprising, prior to transmitting, computing a hash digest using first client-specific data comprising a valid client name together with a key corresponding to the valid client name, and a current time.

6. The process of claim 1, wherein transmitting comprises transmitting the hash digest together with a valid client name corresponding to the hash digest.

7. The process of claim 1, further comprising, prior to transmitting, computing a hash digest using first client-specific data comprising a valid client name together with a key corresponding to the valid client name; and wherein transmitting comprises transmitting the hash digest together with a valid client name corresponding to the hash digest.

8. A process for requesting authentication comprising:

transmitting a hash digest formed from first client-specific data together with second client specific data;

receiving, in response to transmitting, an indication of acceptance when the hash digest and second client-specific data correspond to a valid client authentication request; and

receiving, in response to transmitting, an denial of authentication when the hash digest or the second client-specific data do not correspond to a valid client authentication request.

9. The process of claim 8, further comprising, prior to transmitting, computing the hash digest from first client-specific data and a time-varying function.

10. The process of claim 8, wherein transmitting a hash digest includes transmitting a hash digest computed using first client specific data comprising a valid client name together with a client key.

11. The process of claim 8, wherein transmitting a hash digest comprises transmitting a hash digest formed using an HMAC algorithm.

12. The process of claim 8, wherein transmitting a hash digest comprises transmitting a hash digest computed using first client-specific data comprising a valid client name together with a key corresponding to the valid client name and a current time.

13. The process of claim 8, wherein transmitting a hash digest comprises transmitting a hash digest computed using first client-specific data comprising a valid client name together with a key corresponding to the valid client name, and a current time, and wherein transmitting includes transmitting the current time.

14. The process of claim 8, wherein transmitting comprises transmitting the hash digest together with a valid client name corresponding to the hash digest.

15. The process of claim 8, wherein transmitting a hash digest comprises transmitting a valid client name together with a hash digest computed using and HMAC algorithm and first client-specific data comprising a valid client name together with a key corresponding to the valid client name, and a current time.

16. A process for verification of a client authentication request by a server, comprising:

receiving, in the server, a client authentication request including client-specific data;

comparing the client specific data to data stored in a first cache memory coupled to the server to determine that the client specific data meet a first threshold of validity;

when comparing determines that the client specific data meet the first threshold of validity, proceeding with the authentication process; and

when comparing determines that the client specific data do not meet the first threshold of validity, terminating the verification process.

17. The process of claim 16, further comprising, when comparing determines that the client specific data do not meet the first threshold, storing a portion of the client specific data in a second cache memory along with an indication that the client specific data do not correspond to a valid client.

18. The process of claim 16, wherein:

proceeding with the authentication process comprises second comparing the client specific data with data stored in a second cache memory to determine when the client specific data meet a second threshold of validity and when the client specific data correspond to an identity previously determined to be valid or invalid; and

when the client specific data meet the second threshold, transmitting a request for verification to a database containing client-specific data; and

when the client specific data correspond to an identity previously determined to be invalid, terminating the authentication request.

19. The process of claim 16, wherein receiving comprises receiving data including one or more of: a name, a NameHash, a truncation of a NameHash, a NameKeyHash, a truncation of a NameKeyHash, a TimedNameKeyHash, a truncation of a TimedNameKeyHash or a time.

20. The process of claim 16, wherein receiving comprises receiving a TimedNameKeyHash.

21. The process of claim 16, wherein receiving comprises receiving a TimedNameKeyHash and a current time.

22. The process of claim 16, wherein comparing the client specific data to data stored in a first cache memory comprises comparing a TimedNameKeyHash contained in the authentication request to a function of a stored NameKeyHash and a current time.

23. The process of claim 16, wherein receiving client specific data includes receiving a current time, and further comprising determining when the received current time disagrees with another current time used by the authentication server, and, when the received current time and the another current

time disagree, sending the another current time to an originator of the authentication request.

24. A process for updating a cache memory associated with an authentication server comprising:

sending a request to a database containing information describing authentic users, the request requesting information associated with authentic users that have been entered in the database after a predetermined time;

receiving data corresponding to authentic users where the data have been entered to the database after the predetermined time; and

storing at least a portion of the received data in the cache memory.

25. The process of claim 24, wherein sending a request comprises sending a request for information associated with authentic users that have been added to the database since a previous such request was made.

26. The process of claim 24, wherein receiving comprises receiving a name and a key and the name, and further comprising:

forming a hash digest using the name and a random session key; and

storing client-specific data in the cache memory such that the hash digest may be used as a cachekey to access the client-specific data.

27. The process of claim 24, further comprising computing a hash digest from a valid user name contained in the received data and a random session key stored in the authentication server.

28. The process of claim 24, further comprising:

computing a hash digest from one or more of a valid user name, an associated key and a random session key,

truncating the hash digest; and

storing client-specific data in the cache memory such that the truncated hash digest may be used as a cachekey to access the client-specific data.

29. One or more computer-readable media including instructions that, when executed by one or more processors, causes the one or more processors to:

form an encrypted data string including first client-specific information;

transmit a message including credentials formed using the encrypted data string together with second client-specific information; and

receive an authentication for system access, in response to the message, when the credentials are valid.

30. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string

comprises code configured to cause the one or more processors to form a hash digest from a function of time and a client key.

31. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string comprises code configured to cause the one or more processors to form an encrypted data string from one or more of: a name, a NameHash, a truncation of a NameHash, a NameKeyHash, a truncation of a NameKeyHash, a TimedNameKeyHash, a truncation of a TimedNameKeyHash or a time.

32. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string comprises code configured to cause the one or more processors to form an encrypted data string using a one-way hash function.

33. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string comprises code configured to cause the one or more processors to form a hash digest using an HMAC algorithm.

34. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string comprises code configured to cause the one or more processors to form an encrypted data string using a valid client name and a current time together with a key corresponding to the valid client name.

35. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to form an encrypted data string comprises code configured to cause the one or more processors to form an encrypted data string using a key corresponding to the valid client name and a current time.

36. The computer-readable media of claim 29, wherein the code configured to cause the one or more processors to transmit comprises code that is configured to cause the one or more processors to transmit a plaintext client name as a portion of the second client-specific data.

36. A computer system comprising:

an authentication server; and

a primary cache memory coupled to the authentication server, wherein the authentication server is configured to:

receive a client authentication request including client-specific data;

compare the client specific data to data stored in a first cache memory coupled to the server to determine that the client specific data meet a first threshold of validity;

when comparing determines that the client specific data meet the first threshold of validity, proceed with authentication; and

when comparing determines that the client specific data do not meet the first threshold of validity, terminate authentication and deny the authentication request.

37. The computer system of claim 36, wherein the authentication server is configured to employ a first, plaintext portion of the client-specific data as a cachekey to obtain related encrypted client-specific data from the first cache memory.

38. The computer system of claim 36, wherein the authentication server is further configured to store at least some of the client specific data in a second cache memory along with an indication that the client specific data do not correspond to a valid client when comparing determines that the client specific data do not meet the first threshold.

39. The computer system of claim 36, wherein the authentication server is configured to second compare the client specific data with data stored in a second cache memory to determine when the client specific data meet a second threshold of validity and when the client specific data correspond to an identity previously determined to be valid or invalid;

when the client specific data meet the second threshold, transmit a request for verification to a database containing client-specific data; and

when the client specific data correspond to an identity previously determined to be invalid, terminate the authentication request.

40. The computer system of claim 36, wherein the client-specific data includes a NameKeyHash that is also a function of time.

41. The computer system of claim 36, wherein the client-specific data includes a TimedNameKeyHash.

42. The computer system of claim 36, wherein the client-specific data includes a TimedNameKeyHash and a current time is included with the client-specific data.

43. The computer system of claim 36, wherein the client specific data stored in the first cache memory comprises a NameKeyHash, and wherein the authentication server is configured to form a TimedNameKeyHash from the NameKeyHash and to compare the formed TimedNameKeyHash to a portion of the client-specific data.

44. The computer system of claim 36, wherein the client specific data includes a current time, and wherein the authentication server is further configured to determine when the received current time disagrees with another current time used by the authentication server, and when the received current time and the another current time disagree, send the another current time to an originator of the authentication request.

45. An authentication process comprising:

transmitting, by a user, client specific data including at least one of first client specific data, a client name and a proof of knowledge of a client key; and

receiving, in response to transmitting, an authentication to access a remote computer.

46. An authentication process comprising:

transmitting, by a user, client specific data including at least one of first client specific data, a client name, proof of knowledge of a client key and a NameKeyHash; and

receiving, in response to transmitting, an authentication to access a remote computer.

47. The process of claim 46, wherein transmitting includes transmitting a truncation of the NameKeyHash formed using the client key and client name.

48. An authentication process comprising:

transmitting, by a user, client specific data including at least one of first client specific data, a client name, a TimedNameKeyHash and a current time; and

receiving, in response to transmitting, an authentication to access a remote computer.

49. The process of claim 48, wherein the TimedNameKeyHash is formed using the current time, the client name and the client key.

50. A process for authenticating a user, comprising:

receiving an authentication request including first client specific data comprising at least one of a client name and proof of knowledge of a client key;

computing a NameHash using the received client name and a random session key;

using data corresponding to the NameHash as a cachekey to access first validity threshold data from a first cache memory;

comparing the first validity threshold data to the first client specific data; and

terminating authentication when the first validity threshold data do not match the first client data.

51. The process of claim 50, further comprising, when the first validity data do not match the first client data, storing the client key and a CredentialInvalidFlag in a second cache memory.

52. The process of claim 50, further comprising, when the first validity data do match the first client data, employing the client name as a cachekey to access second client validity data from a second cache memory.

53. The process of claim 50, further comprising, when the first validity data do match the first client data, employing the client name as a cachekey to

access second client validity data from a second cache memory, wherein the second client validity data comprise a stored copy of a client key.

54. The process of claim 50, wherein using data corresponding to the NameHash as a cachekey comprises using a truncation of the NameHash to access first validity threshold data from a first cache memory.

55. A process for authenticating a user, comprising:

receiving an authentication request including at least one of first client specific data comprising at least one of a client name, proof of knowledge of a client key and a NameKeyHash;

computing a NameHash using the received client name and a random session key;

using data corresponding to the NameHash as a cachekey to access first validity threshold data from a first cache memory;

comparing the first validity threshold data to the first client data; and

terminating authentication when the first validity threshold data do not match the first client data.

56. The process of claim 55, further comprising, when the first validity data do not match the first client data, storing the client key and a CredentialInvalidFlag in a second cache memory.

57. The process of claim 55, further comprising, when the first validity data do match the first client data, employing the client name as a cachekey to access second client validity data from a second cache memory.

58. The process of claim 55, further comprising, when the first validity data do match the first client data, employing the client name as a cachekey to access second client validity data from a second cache memory, wherein the second client validity data comprise a stored copy of a client key.

59. The process of claim 55, wherein using data corresponding to the NameHash as a cachekey comprises using a truncation of the NameHash to access first validity threshold data from a first cache memory.